# Peer-to-Peer Networking with Game Kit

## Session 318

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

# Peer-to-Peer Networking with Game Kit

**Joe Abuan**
Interactive Media Group, Software Engineering

# WELCOME

CHINESE CHECKERS
SINGLE PLAYER
龙 MULTIPLAYER 龙
SETTINGS
HELP
MICHAEL
KEVIN
DAIZY
MELISSA

# Agenda

- Overview of capabilities
- The APIs
  - GKSession
  - GKPeerPickerController
  - GKVoiceChatService
- Code samples
- Tips and best practices

# What You Will Learn

- How to use Game Kit to connect your application

Handles Networking

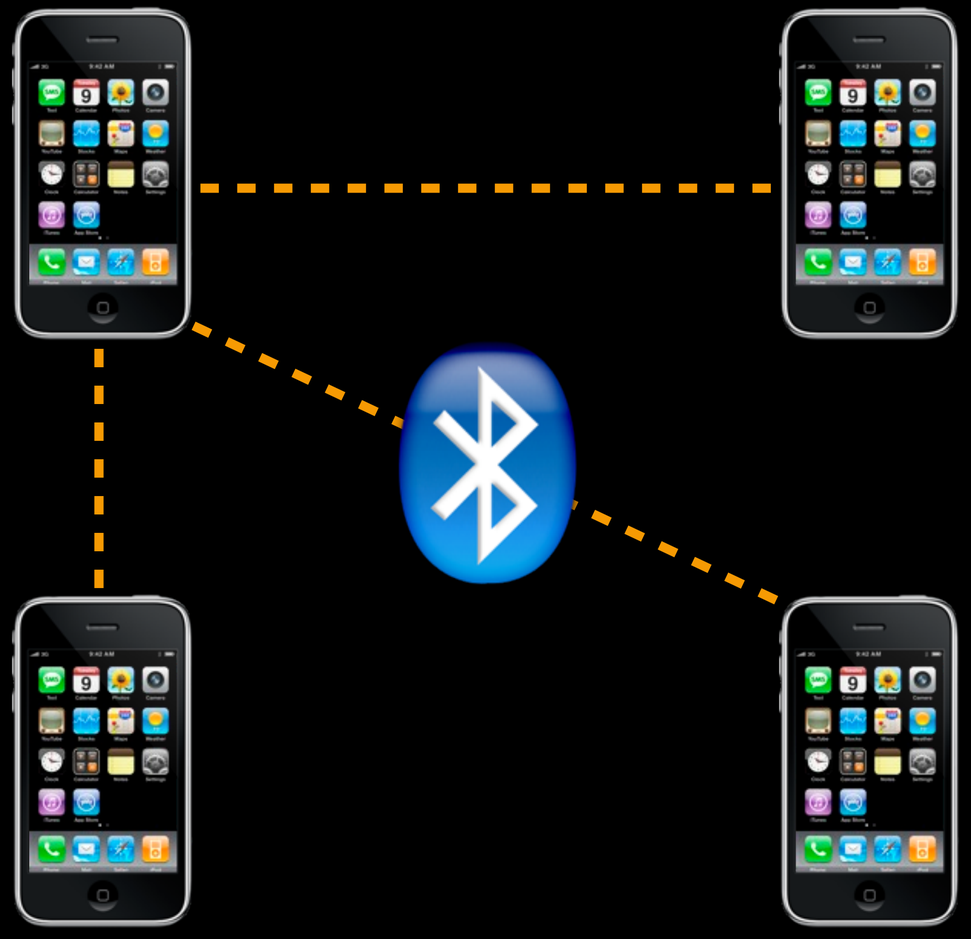
UI for Connecting

# What You Will Learn

Duarte

Need better quality phone icons with a running multi-user application

- How to use Game Kit to connect your application



Bluetooth Support:
iPhone 3G
Second-Gen iPod Touch

# What You Will Learn

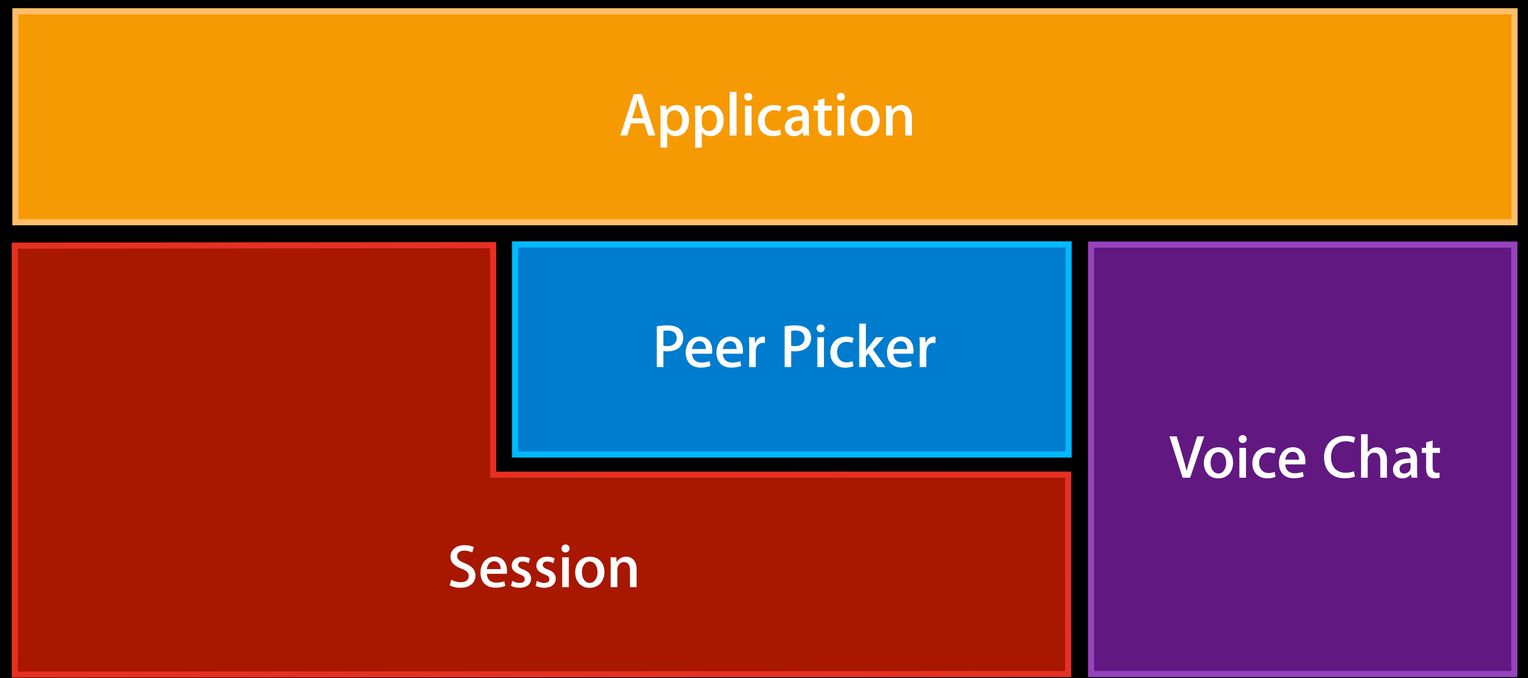
- How to use Game Kit to integrate voice chat

GKVoiceChatService

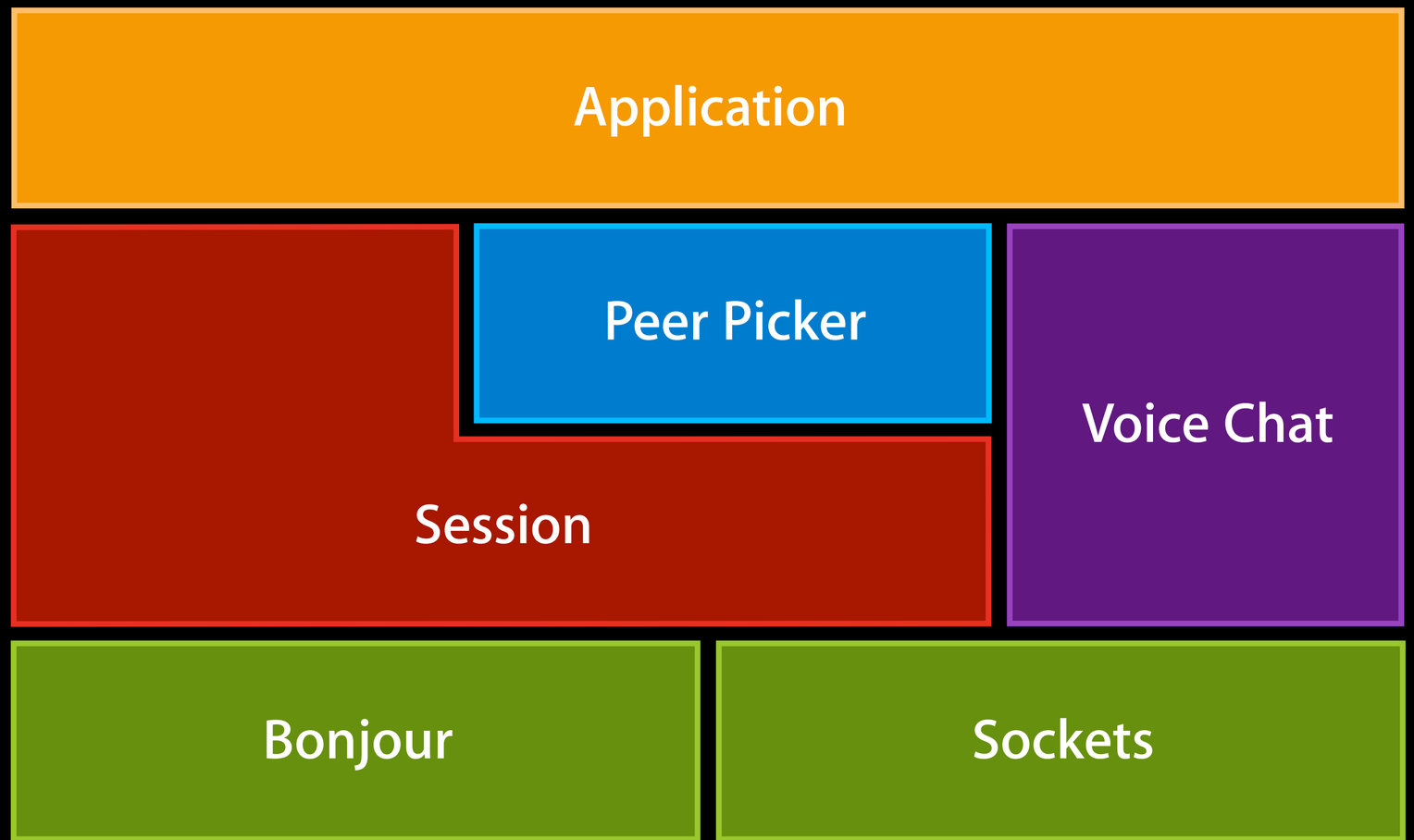
- Integrates with your existing networking code
- Includes echo suppression

# Overview

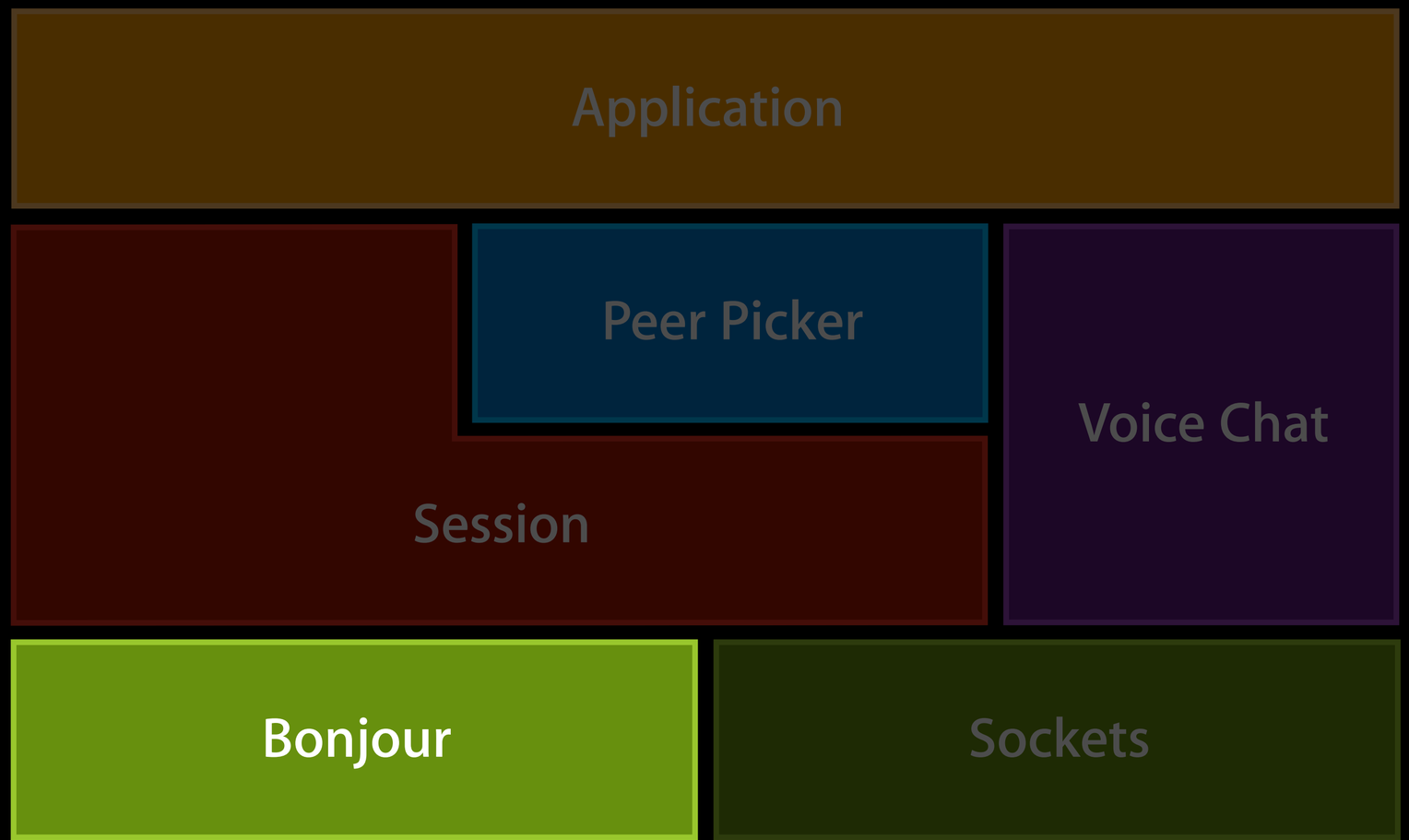
# Overview



# Overview


Application
Peer Picker
Voice Chat
Session
Bonjour
Sockets

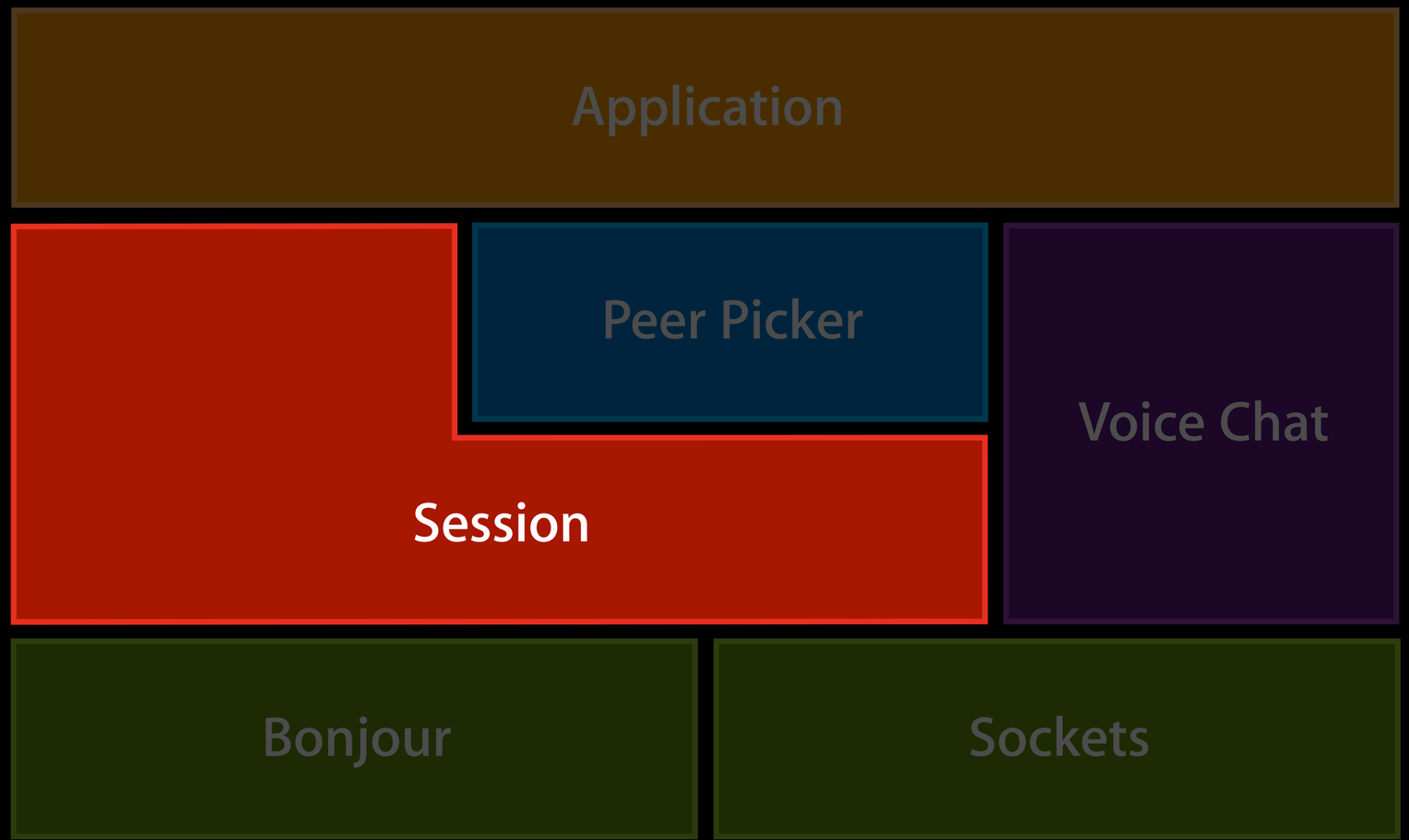
# Overview




Zero Configuration Networking Using Bonjour

Russian Hill
Wednesday 10:30AM

# Overview


Application
Peer Picker
Session
Voice Chat
Bonjour
Sockets

# GKSession Class

- Terminology
  - Peer

# GKSession Class

- Terminology
  - Peer
  - Peer ID

Peer

peerID-1

Peer

peerID-2

Peer

peerID-3

Peer

peerID-4

# GKSession Class

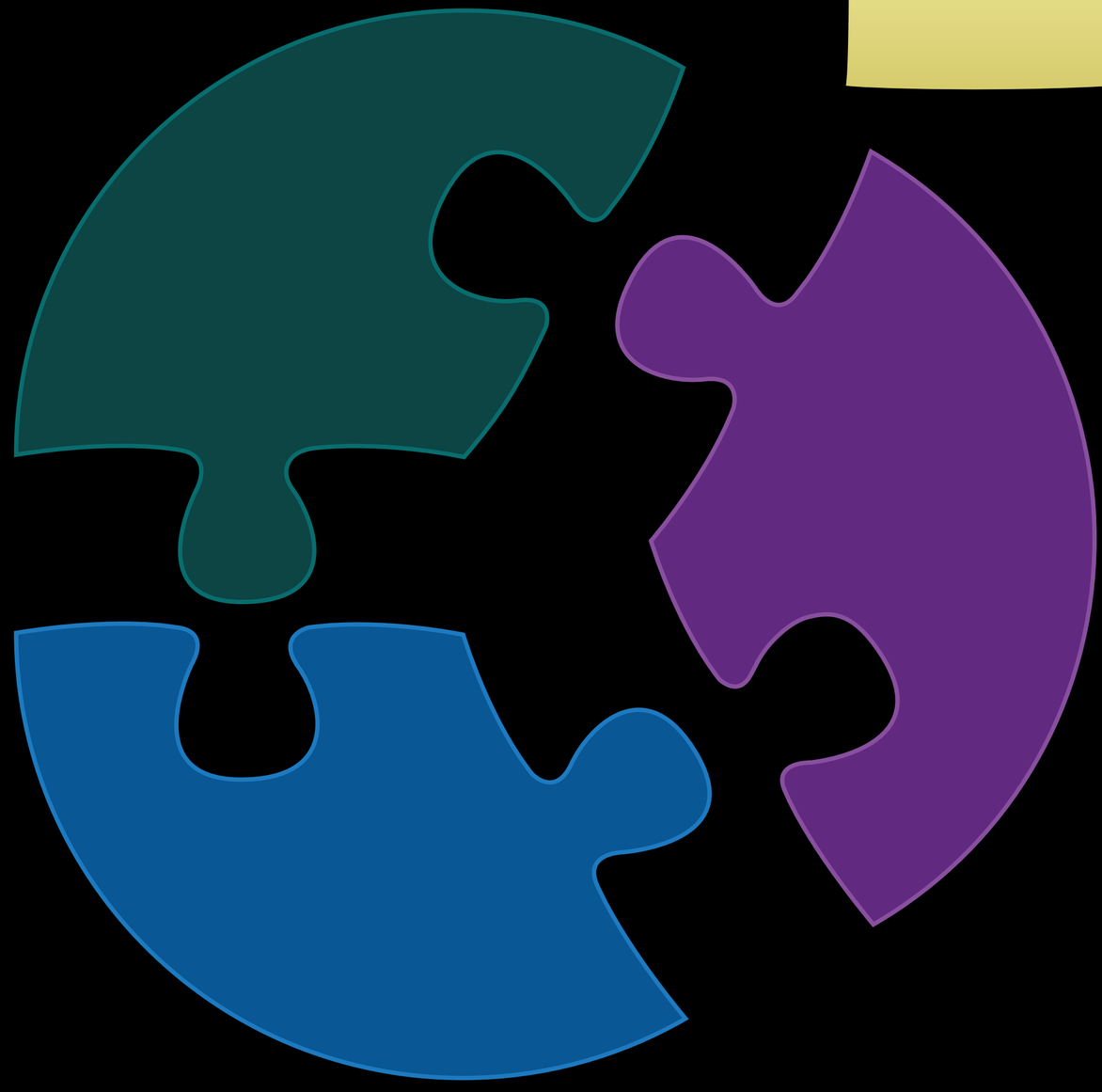
- Terminology
  - Peer
  - Peer ID
  - displayNameForPeerID

# The Three Stages

- Initialize the GKSession
- Connect to other users
- Send and receive data

Duarte

Some nice graphic here with three parts that combine to make one big piece



# The Three Stages

- Initialize the GKSession
- Connect to other users
- Send and receive data

Duarte

Some nice graphic here with three parts that combine to make one big piece

# Stage 1: Initialize the GKSession

- Create the object

```
session = [[GKSession alloc] initWithSessionID:  MY_SESSION_ID
                             displayName:        nil
                             sessionMode:        GKSessionModePeer  ];
```

# Stage 1: Initialize the GKSession

- Create the object

```
session = [[GKSession alloc] initWithSessionID:  MY_SESSION_ID
                             displayName:        nil
                             sessionMode:        GKSessionModePeer  ];
```

- Session ID
  - Uniquely identifies your application
  - Recommended to be a registered Bonjour service type

# Stage 1: Initialize the GKSession

- Create the object

```
session = [[GKSession alloc] initWithSessionID:  MY_SESSION_ID
                             displayName:        nil
                             sessionMode:        GKSessionModePeer  ];
```

- Display name
  - Public name for this session

```
NSString *displayName = [session displayNameForPeer: peerID];
```

# Stage 1: Initialize the GKSession

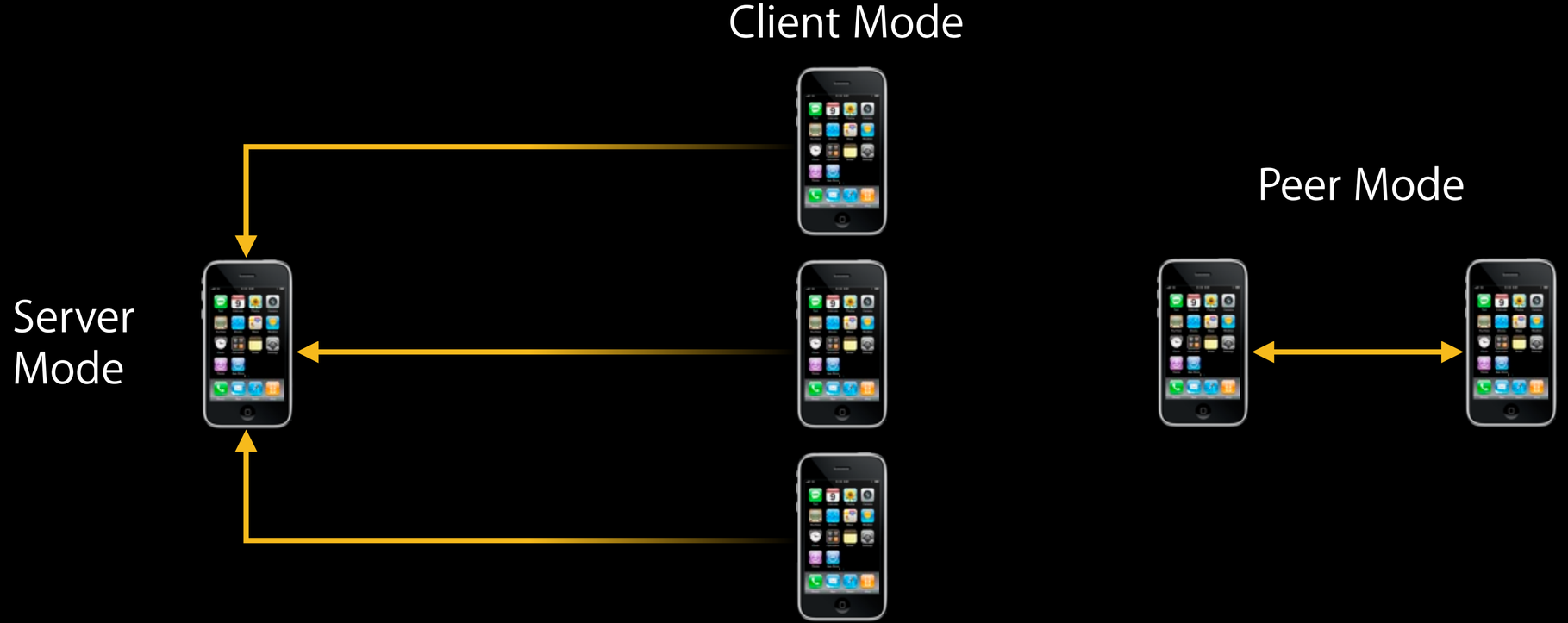
Duarte

Nicer connection arrows

unidirectional
vs
bidirectional (peer)

- Create the object

```
session = [[GKSession alloc] initWithSessionID:  MY_SESSION_ID
                             displayName:        nil
                             sessionMode:        GKSessionModePeer  ];
```



# Stage 1: Initialize the GKSession

- Create the object

```
session = [[GKSession alloc] initWithSessionID:  MY_SESSION_ID
                             displayName:        nil
                             sessionMode:        GKSessionModePeer  ];
```

- Set the delegate

```
session.delegate = self;
```

- Set as available

```
session.available = YES;
```

# Stage 2: Connect to Other Users

- Establish connections

```
// Send connection invitation to a peer
[session connectToPeer: peerID withTimeout: 0];
```

- Watch for incoming connections

```
- (void) session:                                  (GKSession *) session
         didReceiveConnectionRequestFromPeer:  (NSString *)  peerID { }
```

- Respond to connection request

```
// Accept the connection request
[session acceptConnectionFromPeer: peerID error: &error];
```

# Stage 2: Connect to Other Users

- Watch for peer state changes

```
- (void) session:          (GKSession *) session
        peer:              (NSString *)  peerID
        didChangeState:    (GKPeerConnectionState) state { }
```

- Available state

```
if (GKPeerStateAvailable == state) {
    // Found a peer that can accept connection requests
```

- Connected state

```
if (GKPeerStateConnected == state) {
    // Now connected to another peer
```

# Stage 3: Send and Receive Data

- Send data

```
[session sendDataToAllPeers:  appData
          withDataMode:       GKSendDataUnreliable
          error:              &error  ];
```

# Stage 3: Send and Receive Data

- Send data

```
[session sendDataToAllPeers:  appData
          withDataMode:       GKSendDataUnreliable
          error:              &error  ];
```

- Wrap the data you want to send in an NSData object

# Stage 3: Send and Receive Data

- Send data

```
[session sendDataToAllPeers:  appData
          withDataMode:       GKSendDataUnreliable
          error:              &error  ];
```

- Specify the data delivery mode
  - Reliable
  - Unreliable

# Stage 3: Send and Receive Data

- Send data

```
[session sendDataToAllPeers:  appData
          withDataMode:       GKSendDataUnreliable
          error:              &error  ];
```

- Detect transmission error

# Stage 3: Send and Receive Data

- Send data

```
[session sendDataToAllPeers:  appData
          withDataMode:       GKSendDataUnreliable
          error:              &error  ];
```

- Set the data receive handler

```
[session setDataReceiveHandler: self withContext: NULL];
```

- Receive data

```
- (void) receiveData:  (NSData *) data
          fromPeer:    (NSString *) peerID
          inSession:   (GKSession *) session
          context:     (void *) context { }
```

# GKSession Review

- Creates the peer-to-peer network and delivers your data
  - Works over Bluetooth
  - Peers and peer IDs
- Three stages
  - Initialize
  - Connect
  - Deliver data

# Using the Peer Picker

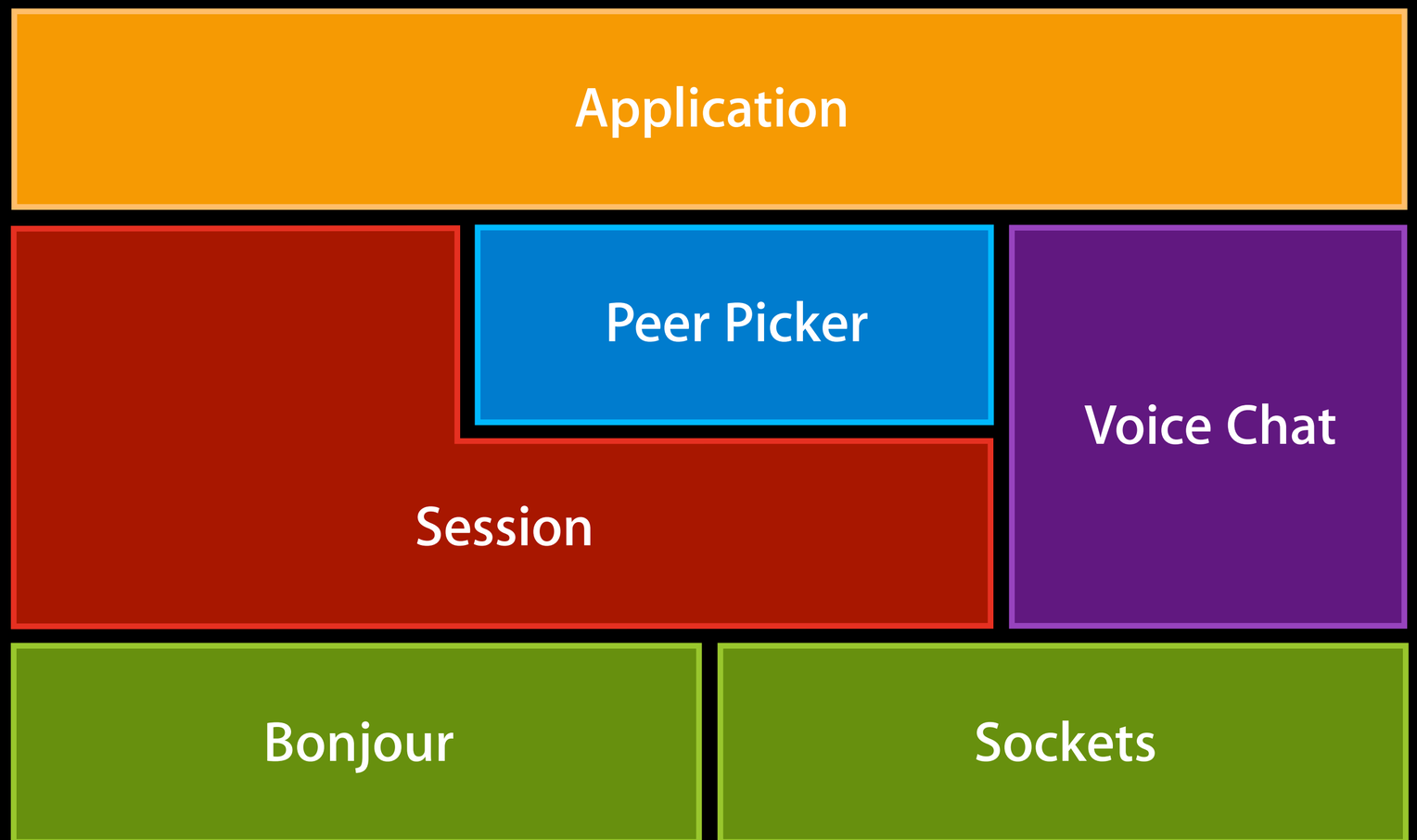
## GKPeerPickerController

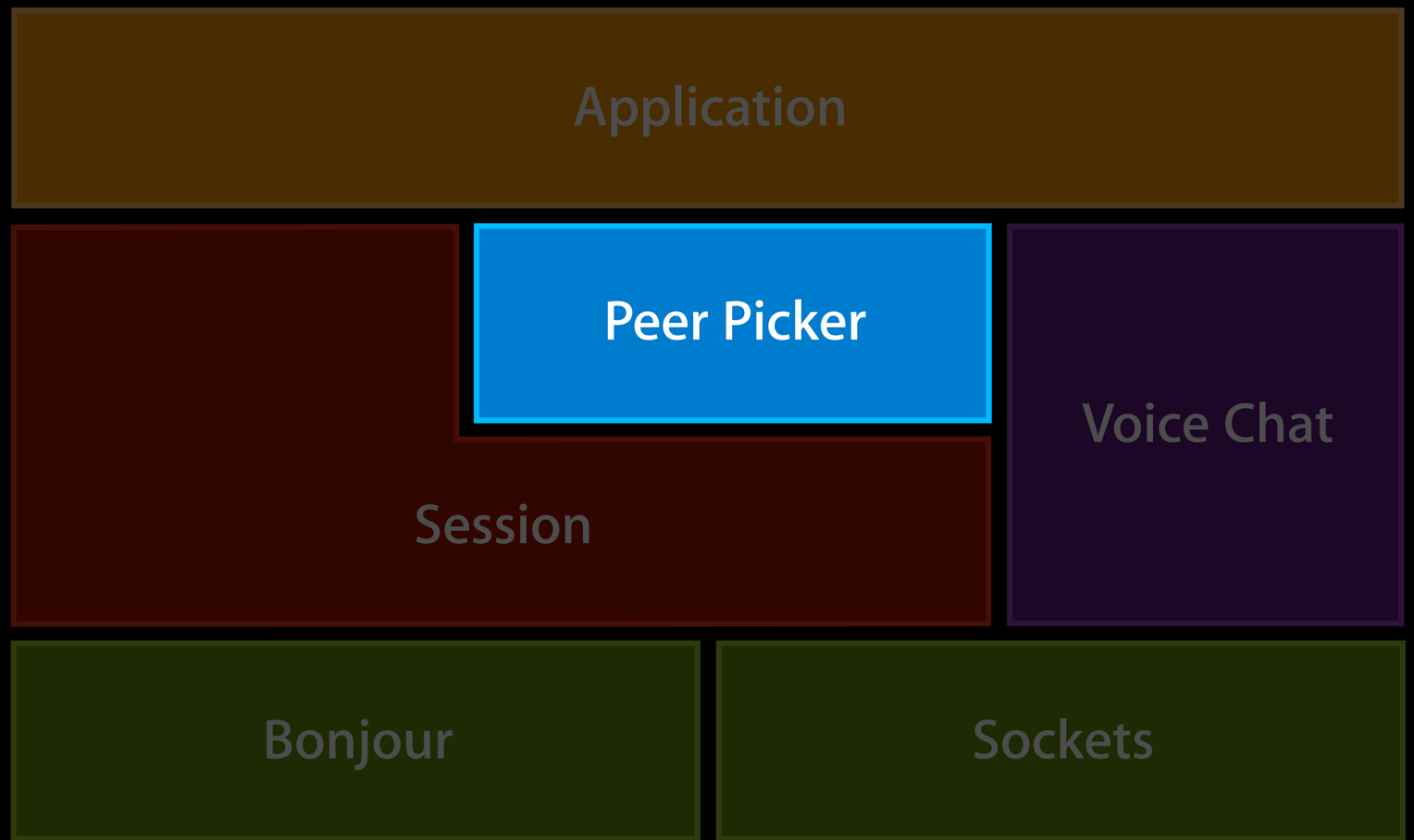
**Jonathan Bennett**
Game Kit Engineer

Standard UI

Connect
Two Peers

# What Is the Peer Picker?


Application
Peer Picker
Session
Voice Chat
Bonjour
Sockets

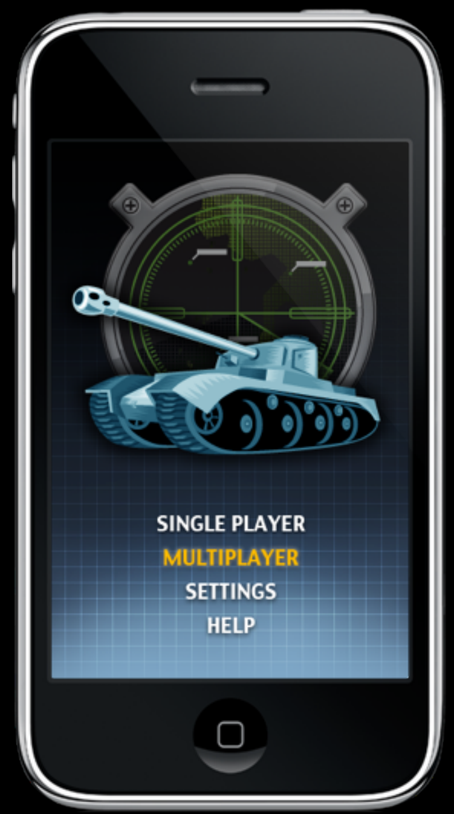

Application
Peer Picker
Session
Voice Chat
Bonjour
Sockets

# What Can Users Do with Peer Picker?

- some real or faked app in background instead of the colored boxes
- iPhone surrounding screenshot
- Note there are multiple images in this smart build

- List nearby discovered iPhones
- Invite another iPhone to connect
- Accept invitations
- Connect to another peer
- Turn on Bluetooth if needed

John's iPhone




Show Peer Picker

Application

Peer Picker

Duarte-

Need a phone with a main menu of an app (same theme as the other mock-ups as the other slides).

Also, any cleanup you see fit of this series of builds over the next few slides would be great.

# John's iPhone

Session?

Application

Peer Picker

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

John's iPhone

Session

John's iPhone

Session

Application

Peer Picker

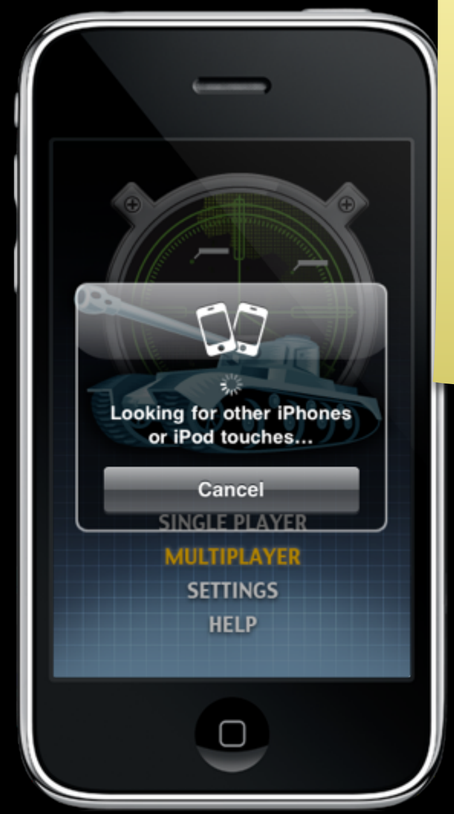
Session

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

John's iPhone

Jane's iPhone




Duarte-

Need a good Wireless fan instead of these handdrawn ones.

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

Application

Peer Picker

Session

Found Peer

Application

Peer Picker

Session

Found Peer

John's iPhone

Jane's iPhone

Duarte-

Need better button press animation (although this one kind of works)

Need a list of peers that has each other.

On John's iPhone should show one item "Jane's iPhone" in the list, on Jane's iPhone should show "John's iPhone" in the list.

Same as ebfore, these screenshots should be wrapped in iphones and a fake game UI behind the picker.

John's iPhone

Jane's iPhone

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

John's iPhone

Jane's iPhone

Duarte-

Need better button press animation (although this one works okay).

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

John's iPhone

Jane's iPhone

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

Connected

Application

Peer Picker

Session

Connected

Application

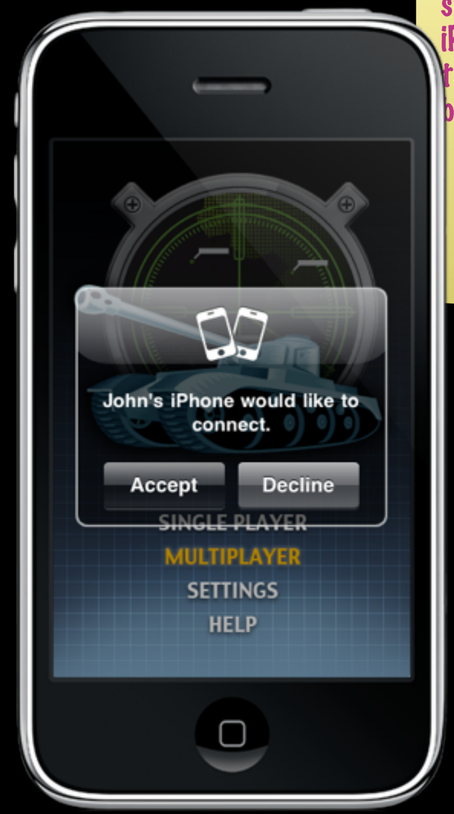
Peer Picker

Session

Connected

John's iPhone

Jane's iPhone




Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

Dismiss

Application

Peer Picker

Session

Connected

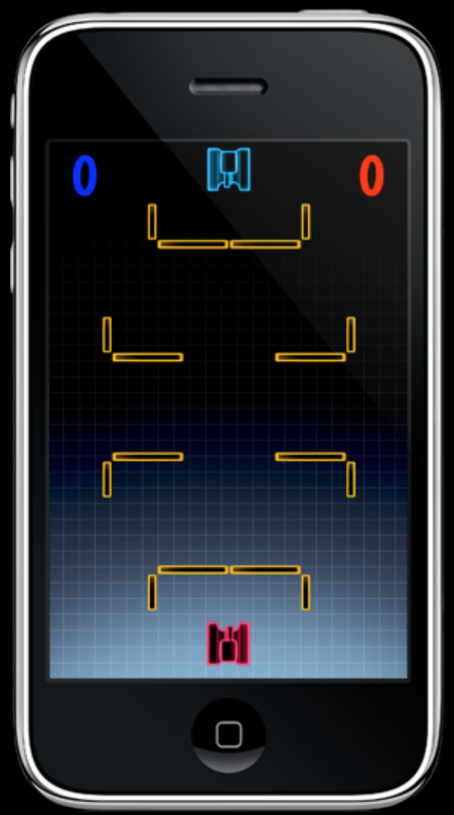
Application

Peer Picker

Session

Dismiss

John's iPhone

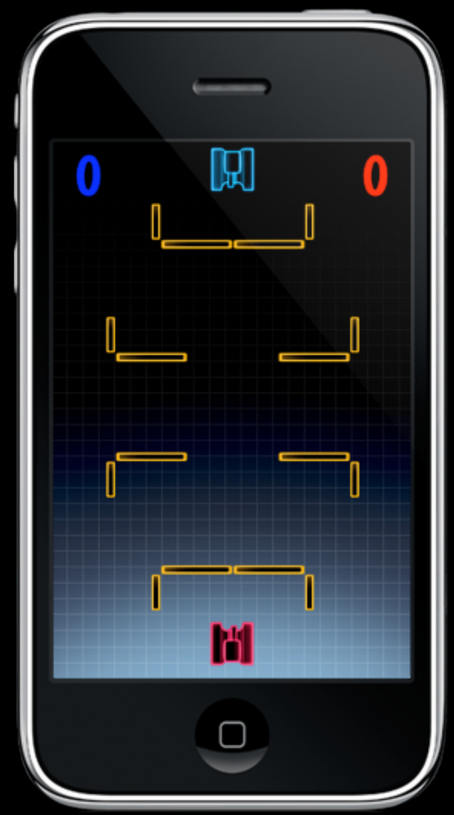



Application

Session

Connected

Jane's iPhone




Application

Session

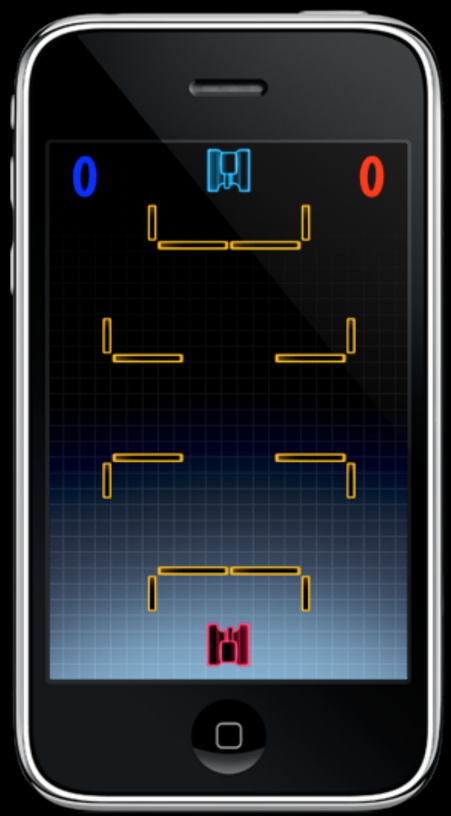
Duarte -

Need image of iphone with in-game graphic (no picker)

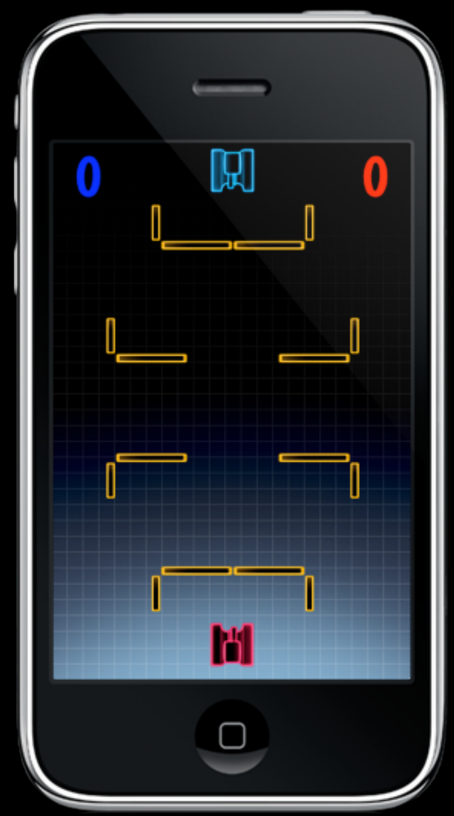
Duarte -

Need image of iphone with in-game graphic (no picker)

John's iPhone




Jane's iPhone




Application

Session

Connected

Application

Session

# Using Peer Picker in Your App

1. Show the Peer Picker to the user
2. Respond to Peer Picker events
3. Start your multiplayer game!

# Using Peer Picker in Your App

- Initialize the Peer Picker

```
GKPeerPickerController *picker = [[GKPeerPickerController alloc] init];
```

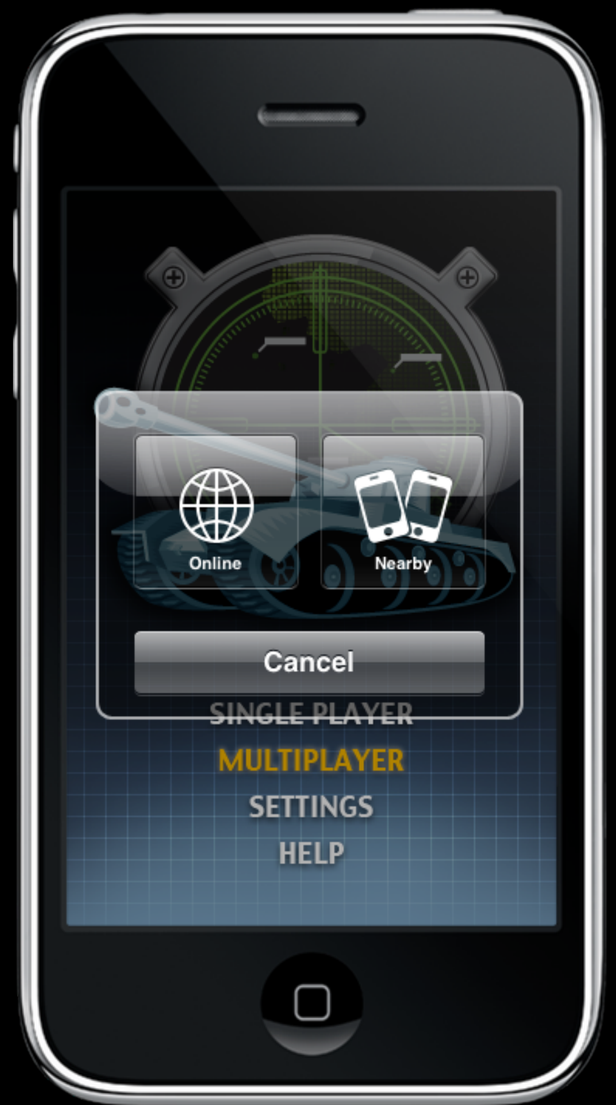
- Set the Peer Picker's delegate

```
picker.delegate = self;
```

- Show the Picker

```
[picker show];
```

# Configuring Connection Types



# Configuring Connection Types

```
GKPeerPickerController *picker = [[GKPeerPickerController alloc] init];
picker.delegate = self;
picker.connectionTypesMask = GKPeerPickerConnectionTypeNearby |
                             GKPeerPickerConnectionTypeOnline;
[picker show];
```

Important: In iPhone OS 3.0, the Peer Picker only configures nearby connections over Bluetooth.  Your application will have to configure other connection types.

# Interacting with Peer Picker

## Peer Picker Delegate

- Responsibilities
  - Respond to user actions
  - Tell Peer Picker who to look for
  - Respond when peers connected

- Delegate adopts the GKPeerPickerControllerDelegate protocol

# Responding to User Cancel

# Responding to User Cancel

# Responding to User Cancel

- Implement -peerPickerControllerDidCancel:
- Called when user dismisses the Peer Picker
- Opportunity to clean up, and show new UI

Don't like this bullet... :-(

```
- (void)peerPickerControllerDidCancel: (GKPeerPickerController *) picker
{
  picker.delegate = nil;
   [picker autorelease];

  if(self.gameSession != nil)
  {
    self.gameSession.available = NO;
    [self.gameSession setDataReceiveHandler:  nil withContext: NULL];
    self.gameSession.delegate = nil;
    self.gameSession = nil;
  }
  // Display main menu UI
}
```

Best sample code?

highlight important parts of sample code...

# Responding to Connection Type Selection

Duarte-

Need better button press (although this one works okay)

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

# Responding to Connection Type Selectio

lots of orange, maybe too much to highlight important parts. consider some type of build to change focus.

- Implement -peerPickerController:didSelectConnectionType:
- Optional: Only needed when supporting multiple connection types
- Dismiss Peer Picker if Online selected

```
- (void) peerPickerController: (GKPeerPickerController *) picker
    didSelectConnectionType: (GKPeerPickerConnectionType) type
{
    if( type == GKPeerPickerConnectionTypeOnline )
    {
    [picker dismiss];
    picker.delegate = nil;
    [picker autorelease];
    // Display your own user interface for configuring Internet connections.
  }
}
```

# Telling Peer Picker Who to Look F

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

Session?

Application

Peer Picker

# Telling Peer Picker Who to Look F

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.

Session

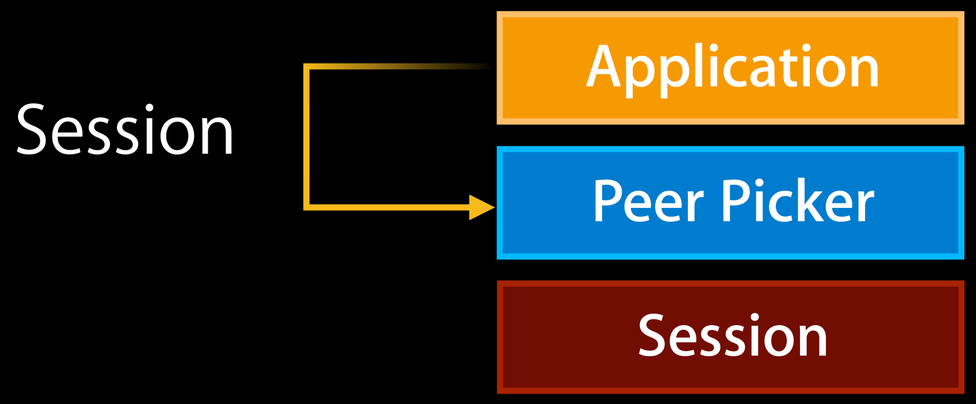
Application

Peer Picker

# Telling Peer Picker Who to Look For

Duarte-

Similar to before, need this screenshot wrapped around an iPhone, with a fake game one-to-one multiplayer type game behind it.



# Telling Peer Picker Who to Look For

I don't like this title :-(

## Providing a GKSession

- Implement -peerPickerController:sessionForConnectionType:
- If not implemented, Peer Picker creates default GKSession object
- Provided GKSession must be configured to GKSessionModePeer

```
- (GKSession *)peerPickerController: (GKPeerPickerController *) picker
        sessionForConnectionType: (GKPeerPickerConnectionType) type
{
  GKSession *session = [[GKSession alloc] initWithSessionID: myExampleSessionID
                                                displayName: playerName
                                                sessionMode: GKSessionModePeer];
    return [session autorelease];
}
```

GKSession for the picker.

Do we need to create other slides to go over some of the different options?

Is this the best correct example? Do we need/want to set the session/delegate here?

Need to highlight important parts of the code.

Look at presenter notes.

# Responding to a Connected Peer

Application

Peer Picker

Session

Accept Invitation

Application

Peer Picker

Session

# Responding to a Connected Peer

Connected

Application

Peer Picker

Session

Connected

Application

Peer Picker

Session

Connected

# Responding to a Connected Peer

- Implement -peerPickerController:didConnectPeer:toSession:
- Retain a reference to the session
- Must dismiss the Peer Picker

```
- (void)peerPickerController: (GKPeerPickerController *) picker didConnectPeer: (NSString *) peerID
                   toSession: (GKSession *) session
{
    self.gameSession = session; // retaining property
    self.gameSession.delegate = self;
    [self.gameSession setDataReceiveHandler: self withContext: NULL];

    [picker dismiss];
    picker.delegate = nil;
    [picker autorelease];
     // start your game.
}
```

# The Peer Picker and Session Relationship

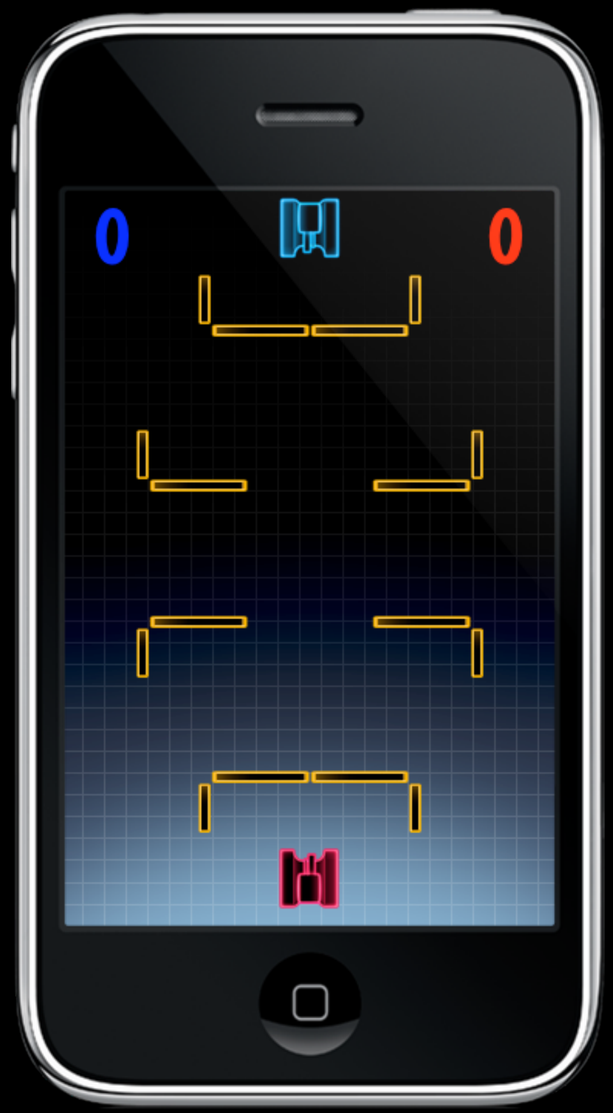
- Peer Picker handles discovery and connection for a session
- Respond to Session events after Peer Picker connects

```
- (void)peerPickerController: (GKPeerPickerController *) picker didConnectPeer: (NSString *) peerID
                   toSession: (GKSession *) session
{
    self.gameSession = session; // retaining property
    self.gameSession.delegate = self;
    [self.gameSession setDataReceiveHandler: self withContext: NULL];
    . . .
}
```

# Start Playing Around!

For Duarte -

Need this wrapped in an iphone




- Sample code: GKTank
- Example two player game
- Demonstrates using Game Kit Peer Picker and Session

# GKVoiceChatService

## Adding voice chat to your application

**Roberto Garcia**
Game Kit Engineer

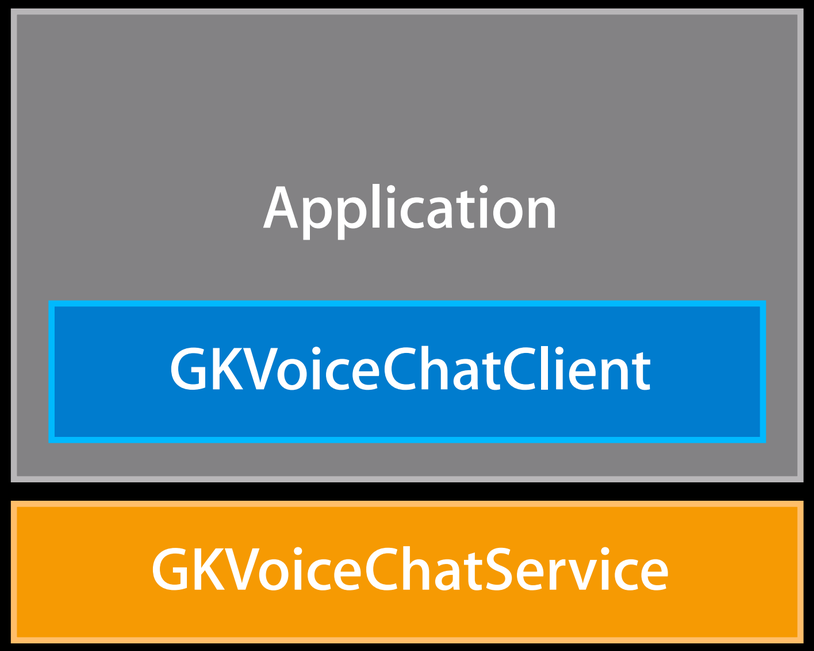
# Overview

- Understanding the API
- Implementation using Bonjour sample code

# Architecture

Duarte

Make sure this color scheme matches previous block color schemes




- Application implements GKVoiceChatClient Protocol
- Calls GKVoiceChatService methods
- GKVoiceChatService uses GKVoiceChatClient protocol to pass setup information

# Using the GKVoiceChatService

- Application calls
  - -(BOOL) startVoiceChatWithParticipantID:
- Remote application instance calls
  - -(BOOL) acceptCallID:error:
- To hangup either instance calls
  - -(void) stopVoiceChatWithParticipantID:

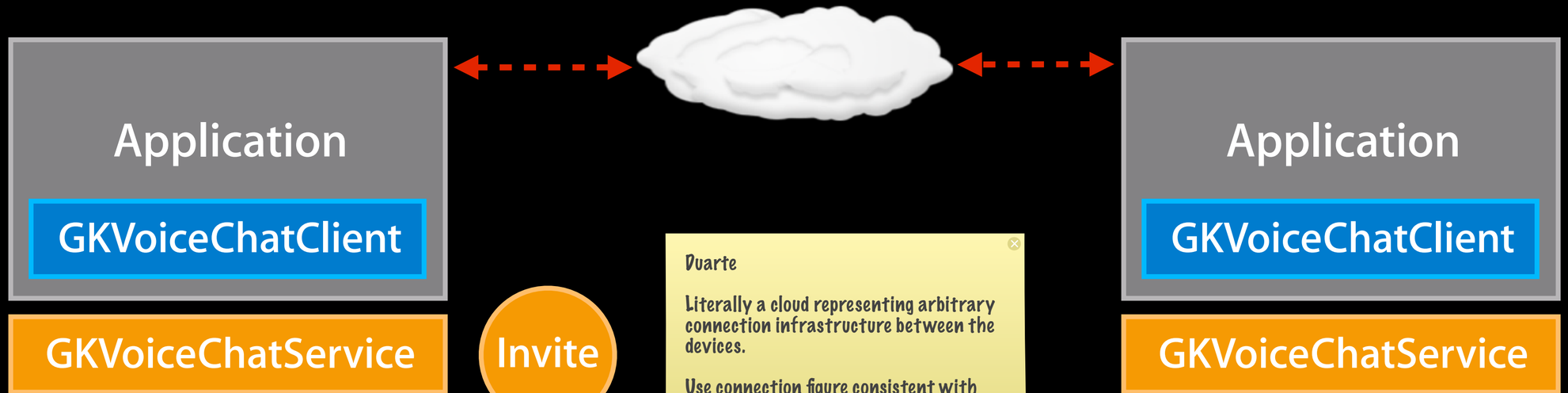
# Voice Chat Setup

Duarte

I use the dissolve effect between slides to animate the disappearance of objects on this slide and the appearance of objects on the next slide.

Duarte

Instant Message Application (like the SMS text application)




Duarte

Literally a cloud representing arbitrary connection infrastructure between the devices.

Use connection figure consistent with previous slides

# Voice Chat Setup

Application

GKVoiceChatClient

GKVoiceChatService

Reply

Application

GKVoiceChatClient

GKVoiceChatService

# Voice Chat Setup

Application

GKVoiceChatClient

GKVoiceChatService

Audio

Application

GKVoiceChatClient

GKVoiceChatService

# Application Provides the Exchange

- GKVoiceChatClient provides channel to exchange setup information for GKVoiceChatService
- GKVoiceChatService drives the exchange!

# GKVoiceChatClient Protocol

- Delegate that provides a channel abstraction
- Only required methods are
  - -(NSString *) participantID
  - -(void) voiceChatService:sendData:toParticipantID
- Must call GKVoiceChatService method
  - -(void) receivedData:fromParticipantID

# -(NSString*)participantID

- An address semantically meaningful only to the client
  - An AIM account name, a fully qualified Jabber ID,  Bonjour Service Instance Name

```
- (NSString *) participantID
{
  return @“bob@fooim.org”;
}
```

# -(void) VoiceChatService:sendData:toParticipantID:

- Called by GKVoiceChatService to send control packets!

```
[voiceChatClient voiceChatService: self
                         sendData: inviteOrReplyData
                   toParticipantID: @“alice@fooim.org”];
```

- Wrap voice chat service packets and send reliably

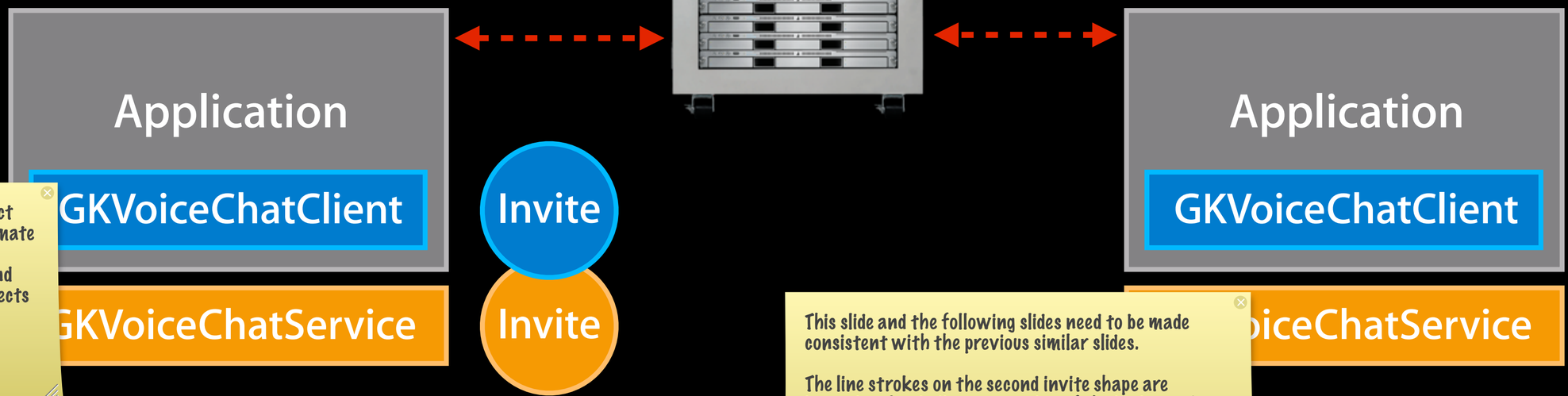
# -(void) VoiceChatService:sendData:toParticipantID:

- Remote instance calls GKVoiceChatService receivedData:fromParticipantID

```
[voiceChatService receivedData:  inviteOrReplyData
             fromParticipantID: @“bob@fooim.org”];
```

# Detailed Voice Chat Setup

startVoiceChat




I use the dissolve effect between slides to animate the disappearance of objects on this slide and the appearance of objects on the next slide.

This slide and the following slides need to be made consistent with the previous similar slides.

The line strokes on the second invite shape are intentional to indicate wrapping of the invite packet by the VoiceChatClient/Application

# Detailed Voice Chat Setup

Application

GKVoiceChatClient

GKVoiceChatService

Invite

Application

GKVoiceChatClient

GKVoiceChatService

receivedInvitation

# Detailed Voice Chat Setup

acceptCall

Application

GKVoiceChatClient

GKVoiceChatService

Reply

Application

GKVoiceChatClient

GKVoiceChatService

# Detailed Voice Chat Setup

Application

GKVoiceChatClient

GKVoiceChatService

Reply

Reply

Application

GKVoiceChatClient

GKVoiceChatService

# Detailed Voice Chat Setup

Application

GKVoiceChatClient

GKVoiceChatService

Audio

Application

GKVoiceChatClient

GKVoiceChatService

didStartVoiceChat

didStartVoiceChat

# WiTap

- Bonjour sample code used to establish tcp connections over a subnet
- Add voice chat

# WiTap

Duarte

Embed in an iPhone.

These images are NOT placeholder images

# WiTap

Duarte

Embed in an iPhone.

These images are NOT placeholder images

# Initialization

```
- (void) setupVoiceChat
{
     voiceChatService = [GKVoiceChatService defaultVoiceChatService];
     voiceChatService.client = self;

    [self setupAudioSession];
}
```

# Initialization

```
- (void) setupVoiceChat
{
    voiceChatService = [GKVoiceChatService defaultVoiceChatService];
    voiceChatService.client = self;

   [self setupAudioSession];
}
```

# Initialization

```
- (void) setupVoiceChat
{
     voiceChatService = [GKVoiceChatService defaultVoiceChatService];
     voiceChatService.client = self;

    [self setupAudioSession];
}
```

# Required GKVoiceChatClientMethods

- Implement required methods

```
- (NSString *) participantID
{
  return serviceName;  //service name, service type, domain
}
```

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
                sendData:(NSData *)data
            toParticipantID:(NSString *)participantID
{
  [self performSelectorOnMainThread:@selector(voiceChatSend:)
                                withObject:data
                             waitUntilDone:NO];
}
```

# Sending Voice Chat Setup Messages

- Modify current send method

```
- (void) send:(const uint8_t)message
{
    WitapMsgHeader msgHeader;
    msgHeader.length = htons(sizeof(uint8_t));
    msgHeader.type = kWitapPacketType;

    [outStream write:(const uint8_t *)&msgHeader maxLength:sizeof(WitapMsgHeader)];
    [outStream write:(const uint8_t *)&message maxLength:sizeof(const uint8_t)];
}
```

# Sending Voice Chat Setup Messages

- Modify current send method

```
- (void) send:(const uint8_t)message
{
      WitapMsgHeader msgHeader;
      msgHeader.length = htons(sizeof(uint8_t));
      msgHeader.type = kWitapPacketType;

      [outStream write:(const uint8_t *)&msgHeader maxLength:sizeof(WitapMsgHeader)];
      [outStream write:(const uint8_t *)&message maxLength:sizeof(const uint8_t)];
}
```

# Sending Voice Chat Setup Messages

- Modify current send method

```
- (void) send:(const uint8_t)message
{
    WitapMsgHeader msgHeader;
    msgHeader.length = htons(sizeof(uint8_t));
    msgHeader.type = kWitapPacketType;

    [outStream write:(const uint8_t *)&msgHeader maxLength:sizeof(WitapMsgHeader)];
    [outStream write:(const uint8_t *)&message maxLength:sizeof(const uint8_t)];
}
```

# Sending Voice Chat Setup Messages

- Modify current send method

```
- (void) send:(const uint8_t)message
{
    WitapMsgHeader msgHeader;
    msgHeader.length = htons(sizeof(uint8_t));
    msgHeader.type = kWitapPacketType;

    [outStream write:(const uint8_t *)&msgHeader maxLength:sizeof(WitapMsgHeader)];
    [outStream write:(const uint8_t *)&message maxLength:sizeof(const uint8_t)];
}
```

# Sending Voice Chat Setup Messages

- Modify current send method

```
- (void) send:(const uint8_t)message
{
    WitapMsgHeader msgHeader;
    msgHeader.length = htons(sizeof(uint8_t));
    msgHeader.type = kWitapPacketType;

    [outStream write:(const uint8_t *)&msgHeader maxLength:sizeof(WitapMsgHeader)];
    [outStream write:(const uint8_t *)&message maxLength:sizeof(const uint8_t)];
}
```

# Sending Voice Chat Setup Messages

- Implement voiceChatSend

```
- (void) voiceChatSend:(NSData *)data
{
    ....//checks for stream validity
     WitapMsgHeader msgHeader;
     msgHeader.length = htons([data length]);
     msgHeader.type = kVoicePacketType;

     [outStream write:(const uint8_t *)&msgHeader maxLength:sizeof(WitapMsgHeader)];
     [outStream write:(const uint8_t *)[data bytes] maxLength:[data length]];
    ...
}
```

# Receiving Voice Chat Setup Messages

- Modifying Data Receive

```
NSStreamEventHasBytesAvailable:
//read from the stream
len = [inStream read:(uint8_t *)buffer + maxLength:sizeof(WitapMsgHeader)];

WitapMsgHeader * witapMsgHeader = (WitapMsgHeader *) buffer;
witapMsgHeader->length = ntohs(witapMsgHeader->length)
    ....//handle this case
    } else {
        //We received a remote tap update, forward it to the appropriate
```

# Receiving Voice Chat Setup Messages

- Receiving Data Continued

```
[inStream read:(uint8_t *)buffer  maxLength: witapMsgHeader->length];
if(witapMsgHeader->type ==  kVoicePacketType)
{

NSData *vcData = [NSData dataWithBytes: buffer numBytes: witapMsgHeader->length];
  [voiceChatService receivedData: vcData fromParticipantID: remoteInstanceName];

}else{
 //We received a remote tap update, forward it to the appropriate view
  ....
}
```

# Starting the Voice Chat

- Calling startVoiceChat

```
case NSStreamEventOpenCompleted:
{
    //Normal WiTap setup procedures
    ...
    if(!didStartVoiceChat){
        NSError *error = nil;
        didStartVoiceChat =
                [voiceChatService startVoiceChatWithParticipantID:remoteInstanceName
                error:&error];
        if(didStartVoiceChat){
            //Do something with the UI
        }else{
            //Process error
         }
}
```

# Accepting the Voice Chat

- Accepting the chat

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
didReceiveInvitationFromParticipantID:(NSString *)participantID
                                callID:(NSInteger)callID
{
  NSError *error = nil;

    //we auto-accept here
    didStartVoiceChat = [voiceChatService acceptCallID:callID error:&error];
}
```

# What's Missing from This Code

- AudioSession setup to handle audio interruption (phone calls)
- AudioSession setup to ensure that the audio plays over the speaker if you are not wearing headphones
- Nonblocking NSStreamEventHasBytesAvailable implementation
- Sample code will have this

# Recap

- GKVoiceChatClient exchanges Invite/Reply messages for the GKVoiceChatService layer
- Voice chat is fun, go for it!

# More Information

**Allan Schaffer**
Graphics Technology Evangelist
aschaffer@apple.com

# Related Sessions

| | |
|---|---|
| Zero Configuration Networking Using Bonjour | Russian Hill<br>Wednesday 10:30AM |

# Labs

| | |
|---|---|
| Game Kit | iPhone Lab D<br>Thursday 9:00AM |
| Game Development for iPhone | iPhone Lab C<br>Thursday 9:00AM |

# Q&A

## WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09